National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Mohammed Zubair Shaik
X23228946

School of Computing
National College of Ireland

National College of Ireland
Project Submission Sheet
School of Computing

Student Name	Mohammed Zubair Shaik
Student ID	23228946
Programme	MSc in Cloud Computing
Year	2024-2025
Module	MSc in Research Computing
Supervisor	Yasantha Samarawickrama
Submission Due Date	11/08/2025
Project Title	Cloud-Based Intrusion Detection using Super Learner Ensemble in ICS/SCADA
Word Count	712
Page Count	16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Mohammed Zuabir
Date:	11/08/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Cloud-Based Intrusion Detection using Super Learner Ensemble in ICS/SCADA

Mohammed Zubair Shaik
Student ID: x23228946

1. Introduction

This project develops a machine learning-based Intrusion Detection System (IDS) for ICS/SCADA networks using the ToN-IoT dataset, which includes telemetry, audit logs, and network traffic data under normal and attack scenarios. The pipeline includes data preprocessing, visualization, model training (Random Forest, Gradient Boosting, Naïve Bayes, Logistic Regression, Super Learner), evaluation, and optional cloud deployment using AWS services.

2. Dataset Information

Dataset Link: https://research.unsw.edu.au/projects/toniot-datasets

he datasets can be used for validating and testing various Cybersecurity applications-based AI such as intrusion detection systems, threat intelligence, malware detection, fraud detection, privacy-preservation, digital forensics, adversarial machine learning, and threat hunting.

The datasets have been called 'ToN_IoT' as they include heterogeneous data sources collected from Telemetry datasets of IoT and IIoT sensors, Operating systems datasets of Windows 7 and 10 as well as Ubuntu 14 and 18 TLS and Network traffic datasets. The datasets were collected from a realistic and large-scale network designed at the Cyber Range and IoT Labs, the School of Engineering and Information technology (SEIT), UNSW Canberra @ the Australian Defence Force Academy (ADFA). A new testbed network was created for the industry 4.0 network that includes IoT and IIoT networks. The testbed was deployed using multiple virtual machines and hosts of windows, Linux and Kali operating systems to manage the interconnection between the three layers of IoT, Cloud and Edge/Fog systems. Various attacking techniques, such as DoS, DDoS and ransomware, against web applications, IoT gateways and computer systems across the IoT/IIoT network. The datasets were gathered in a parallel processing to collect several normal and cyber-attack events from network traffic, Windows audit traces, Linux audit traces, and telemetry data of IoT services.

3. System Configuration

3.1 Software Specification

a.Python==3.10.12:

Python is an interpreted , high-level programming language . Its high-level built in datastructures , combined with dynamic typing and dynamic binding , make it very attractive for Rapid Application Development , as well as for use as a scripting or glue language to connect existing components together. Its language concept and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects

b) Visual Studio Code:-

Visual Studio Code (famously known as VS Code) is a free open source text editor by
Microsoft. VS Code is available for Windows, Linux, and macOS.(Visual Studio Code -Code Editing. Redefined)

c) Google Colab:-

Collaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use,while providing access free of charge to computing resources including GPUs.(colab.google,)

3.2 Hardware Specification

1. Operating system: Windows 11
2. Processor: > 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 3.00 GHz
3. System Compatibility: 64-bit
4. Hard Disk: 512GB
5. RAM: 16 GB

4.Libraries Configuration

1. Pandas

2. Numpy

3. Matplotlib

4. Seaborn

5. Plotly

6. Scikit-learn

7. Tensorflow

9. Keras

5. Project Implementation

5.1 Cloud Environment Setup

- **Cloud9**: Configured as the main development IDE.
- **EC2**: Deployed for running and testing the implementation.
- **S3 Bucket**: Created and configured for storing datasets and models.

5.2 Data Collection

- 150 CSV files were collected and stored in the S3 bucket.
- Loaded into the Cloud9 environment using boto3 and pandas.

5.3 Data Preparation

- Cleaned data by handling missing values.
- Removed irrelevant columns.
- Ensured data consistency.

5.4 Data Preprocessing

- Handled missing values in the dataset.
- Encoded categorical variables for model compatibility.
- Normalized numerical features for consistent scaling.
- Removed irrelevant or redundant columns.

5.5 Model Training

Implemented and trained multiple models:

- Random Forest
- Gradient Boosting
- Naïve Bayes
- Logistic Regression
- Super Learner ensemble

Split dataset into 80% training and 20% testing sets**.**

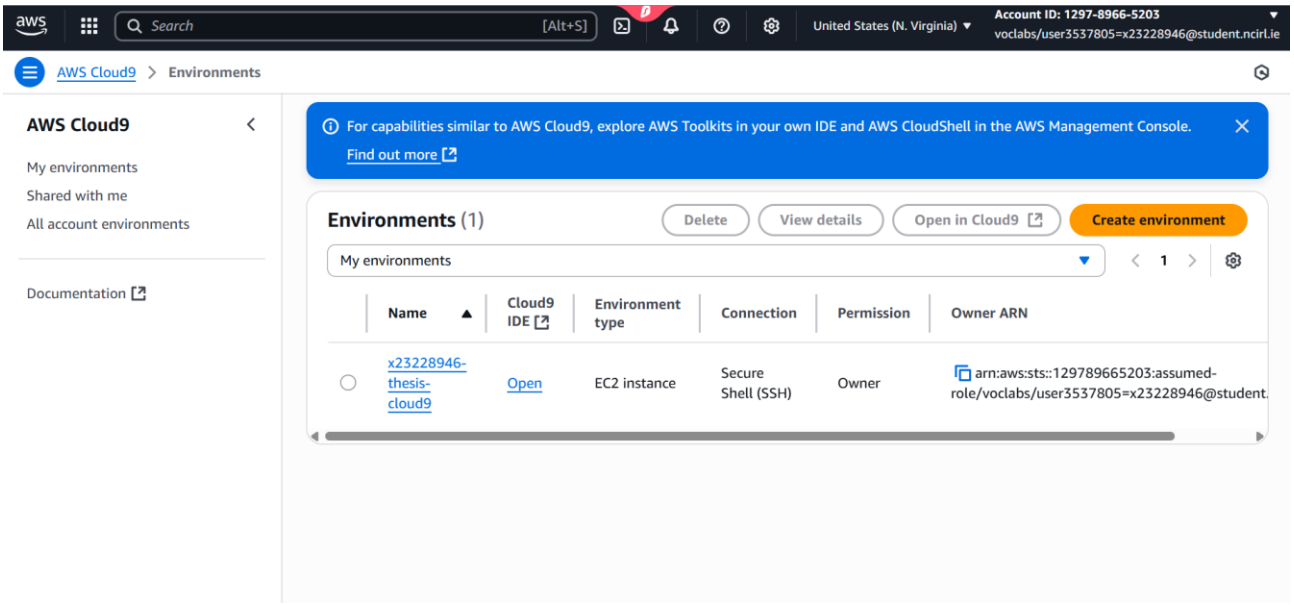
5.6 Performance Evaluation

- Metrics Used: Accuracy, Precision, Recall, F1-score, ROC-AUC.

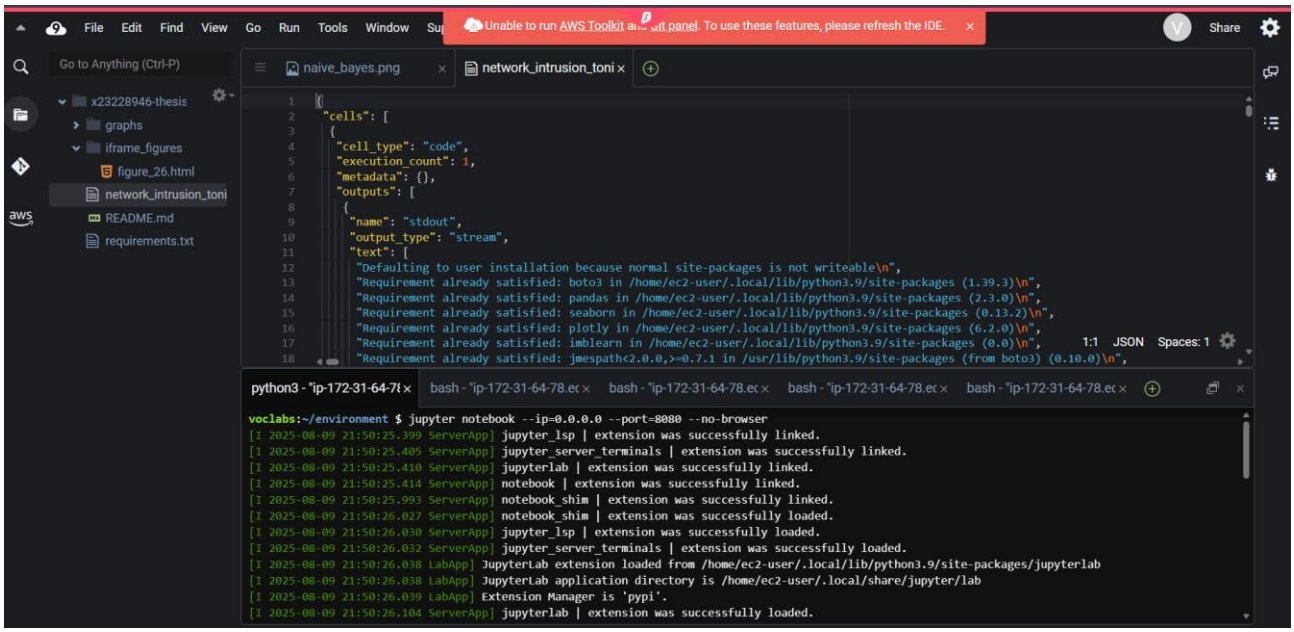
- Generated confusion matrices and classification reports for each model.

6. Visual References

Figure 1: Cloud9 Setup



Cloud9 Enviromen



Now we need to use URL for accessing the jupyter Notebook

Instead of 127.0.0.1 we need to use EC2 Public IPv4 address i.e., 3.238.106.93

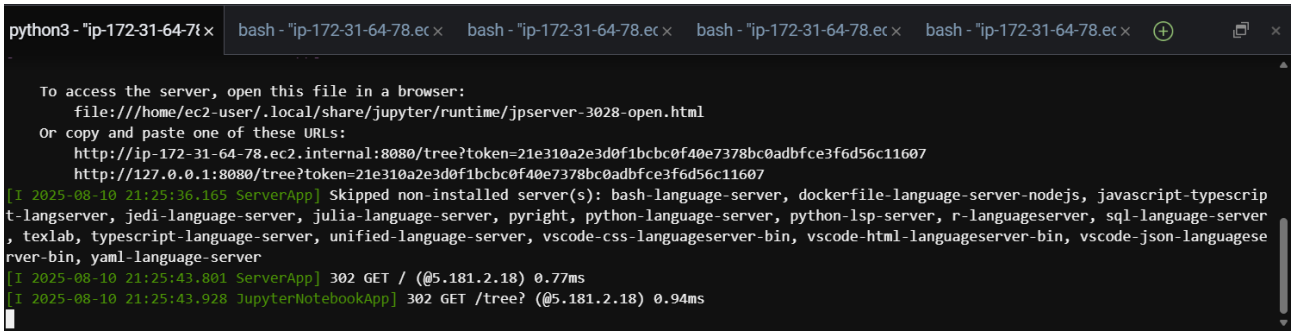
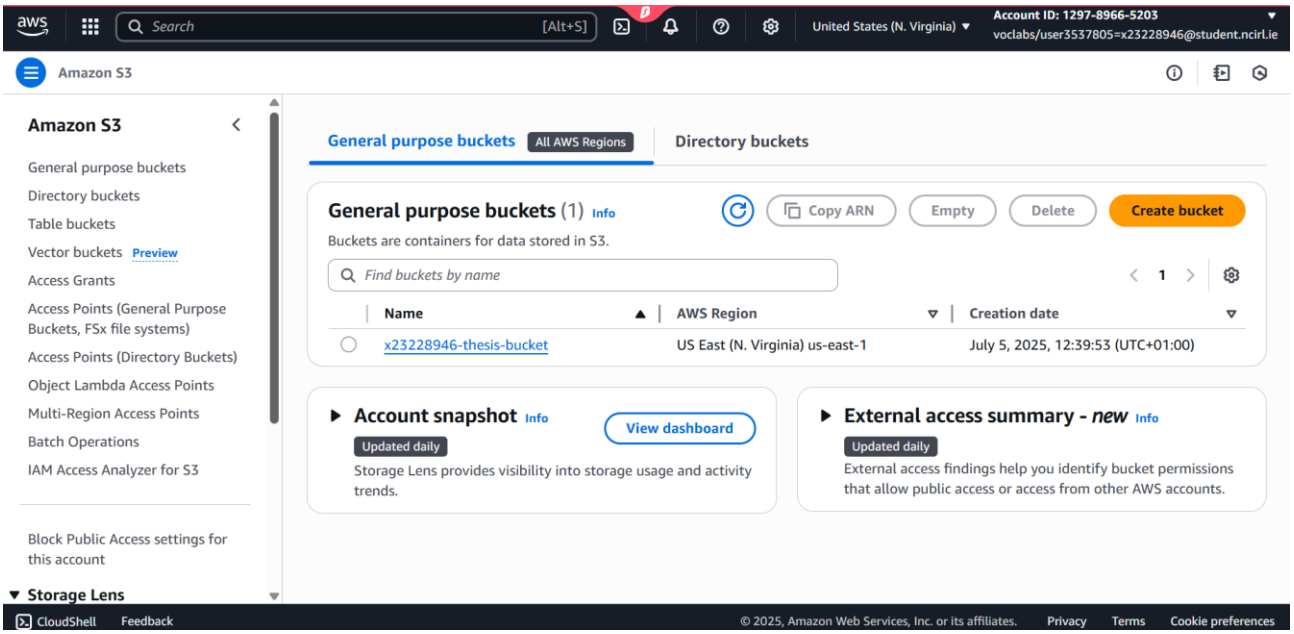
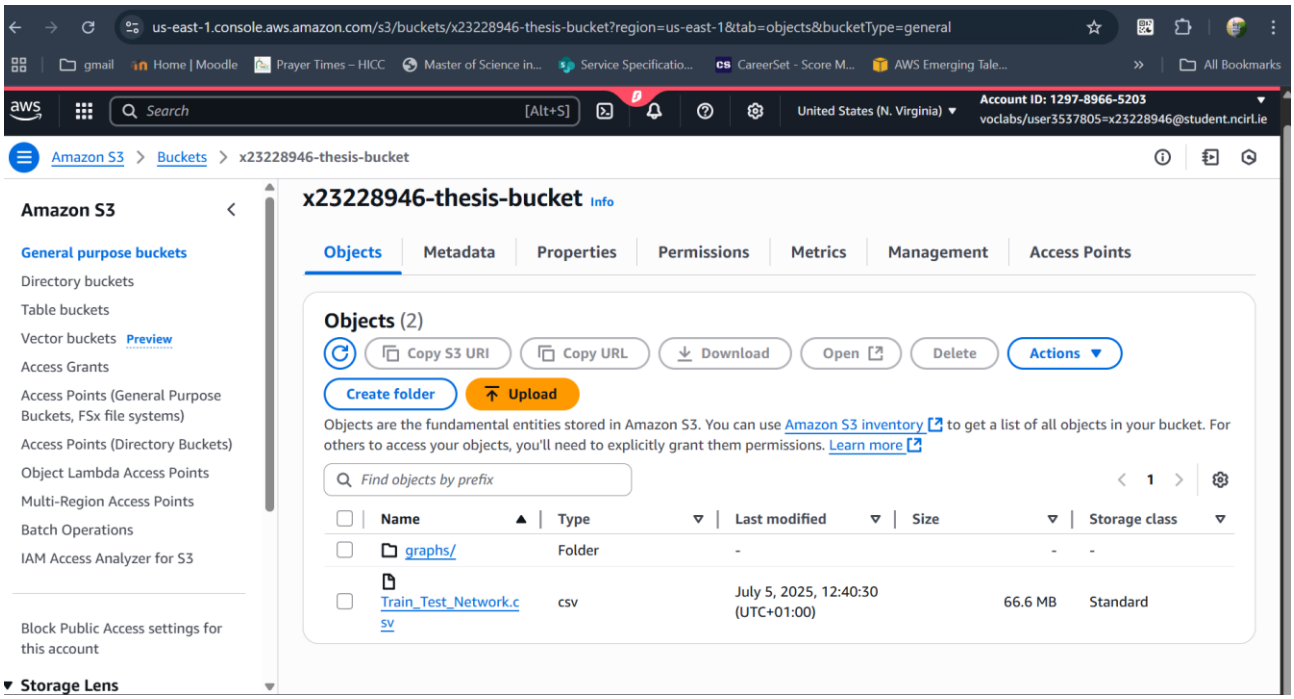


Figure 2: S3 Bucket Configuration


Search [Alt+S]
United States (N. Virginia)
Account ID: 1297-8966-5203
voclabs/user3537805=x23228946@student.ncirl.ie
Amazon S3
Amazon S3
General purpose buckets
Directory buckets
Table buckets
Vector buckets Preview
Access Grants
Access Points (General Purpose Buckets, FSx file systems)
Access Points (Directory Buckets)
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
General purpose buckets All AWS Regions
Directory buckets
General purpose buckets (1) Info
Copy ARN
Empty
Delete
Create bucket
Buckets are containers for data stored in S3.
Find buckets by name
Name
AWS Region
Creation date
x23228946-thesis-bucket
US East (N. Virginia) us-east-1
July 5, 2025, 12:39:53 (UTC+01:00)
Account snapshot Info
Updated daily
View dashboard
Storage Lens provides visibility into storage usage and activity trends.
External access summary - new Info
Updated daily
External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


us-east-1.console.aws.amazon.com/s3/buckets/x23228946-thesis-bucket?region=us-east-1&tab=objects&bucketType=general
Search [Alt+S]
United States (N. Virginia)
Account ID: 1297-8966-5203
voclabs/user3537805=x23228946@student.ncirl.ie
Amazon S3 > Buckets > x23228946-thesis-bucket
Amazon S3
General purpose buckets
Directory buckets
Table buckets
Vector buckets Preview
Access Grants
Access Points (General Purpose Buckets, FSx file systems)
Access Points (Directory Buckets)
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
x23228946-thesis-bucket Info
Objects
Metadata
Properties
Permissions
Metrics
Management
Access Points
Objects (2)
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
Size
Storage class
graphs/
Folder
Train_Test_Network.csv
csv
July 5, 2025, 12:40:30 (UTC+01:00)
66.6 MB
Standard

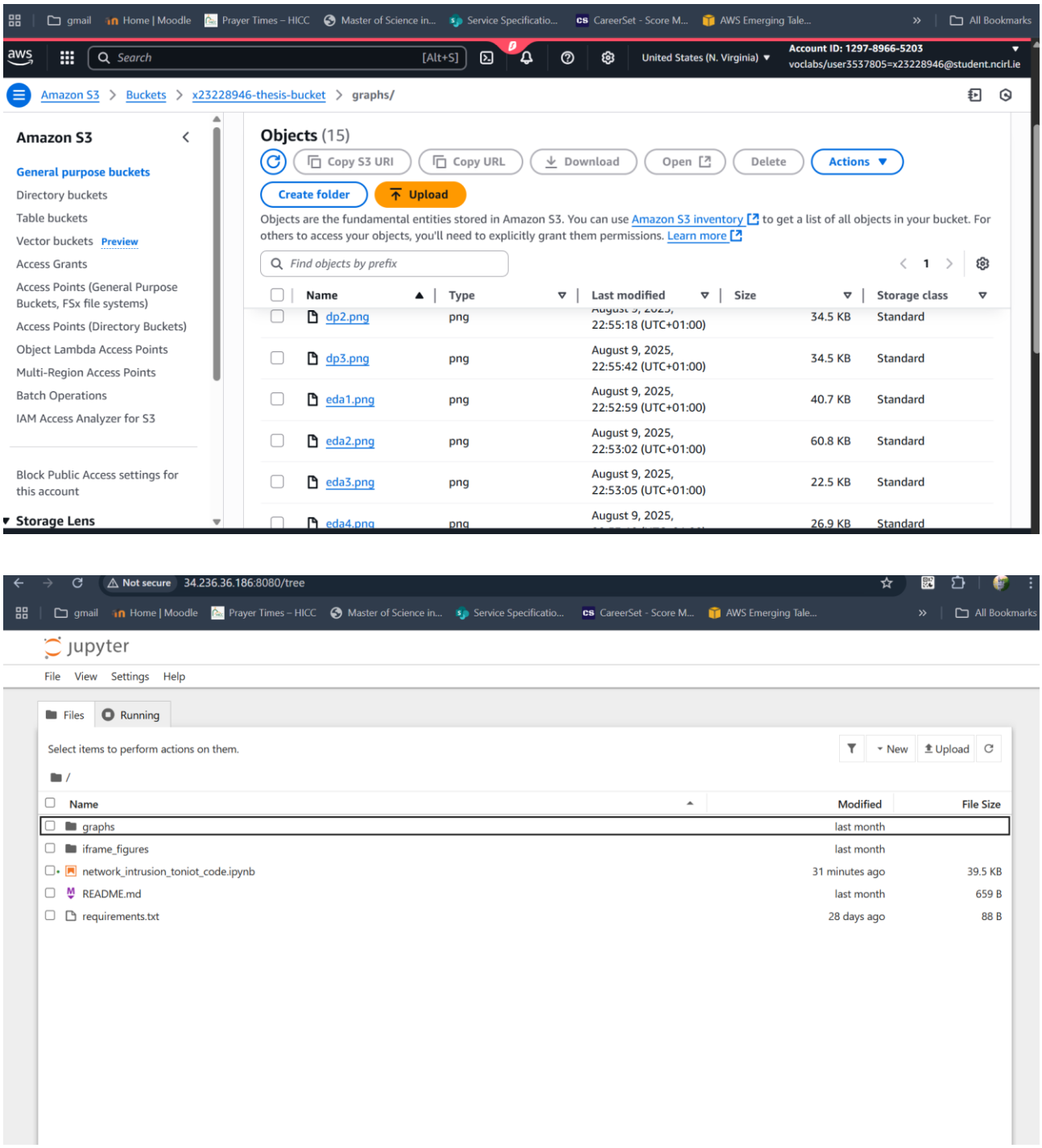
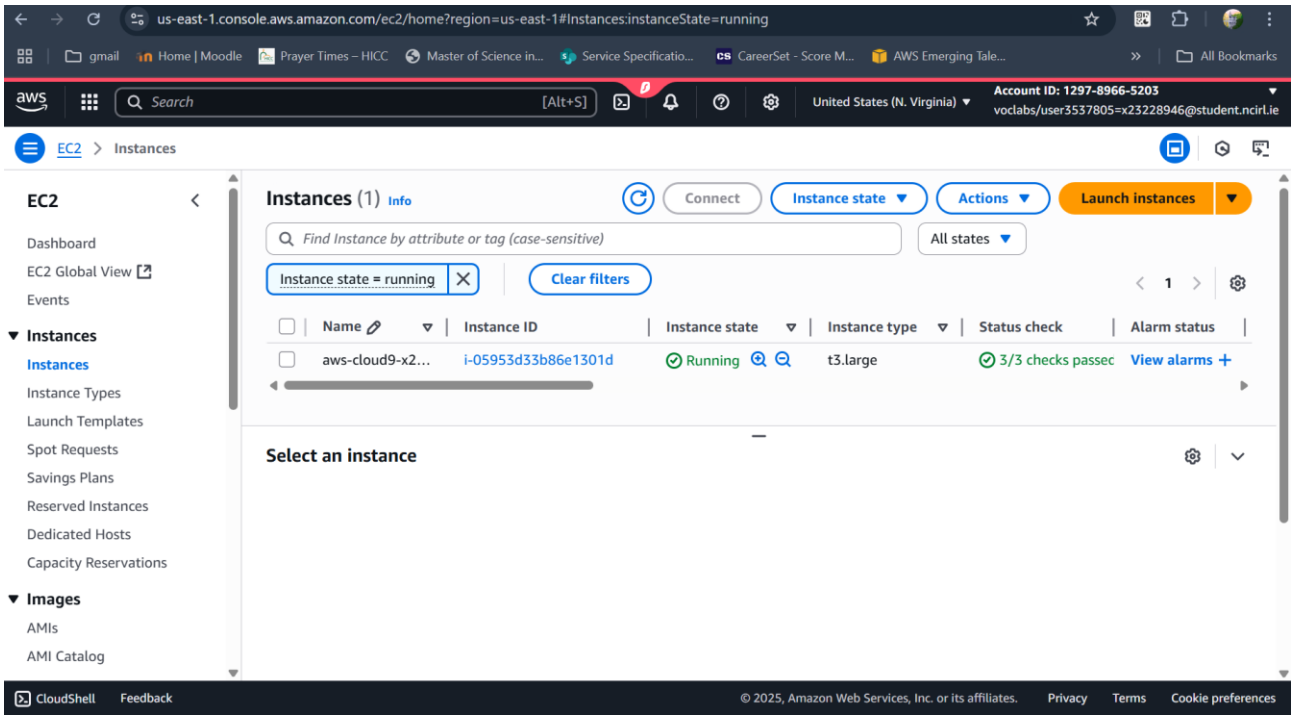


Figure 3: Ec2 Configuration



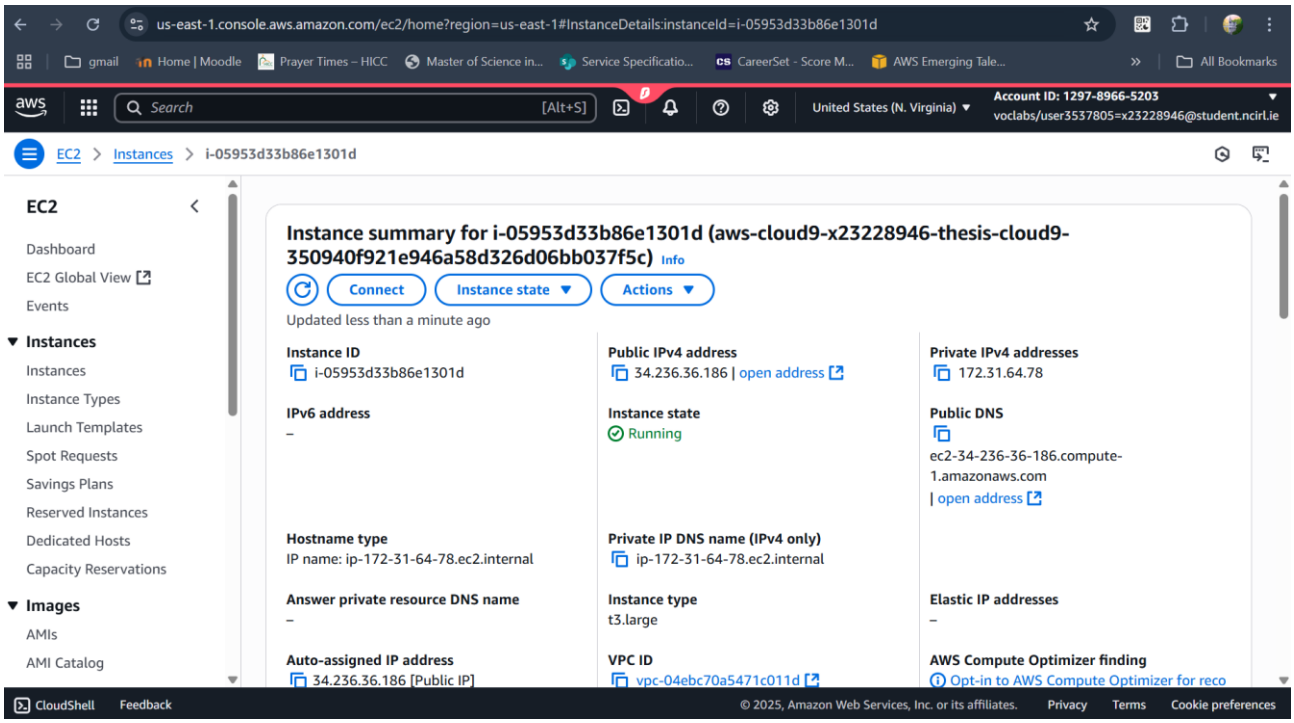


Figure 3: Data Loading Snippet

```
#import all library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import pickle
import plotly.graph_objects as go
import plotly.express as px
import plotly.figure_factory as ff
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn import metrics
from imblearn.under_sampling import NearMiss
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
import warnings
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from lightgbm import LGBMClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from mlxtend.classifier import StackingClassifier
# from pycaret.classification import *
```

Loading dataset from s3 bucket

```
# Config
bucket_name = 'x23228946-thesis-bucket'
file_key = 'Train_Test_Network.csv'

# Create S3 client
s3_client = boto3.client('s3')

# Download CSV file content to memory
response = s3_client.get_object(Bucket=bucket_name, Key=file_key)
csv_content = response['Body'].read().decode('utf-8')

# Load into DataFrame
iotdataset = pd.read_csv(StringIO(csv_content))
```

Random Forest

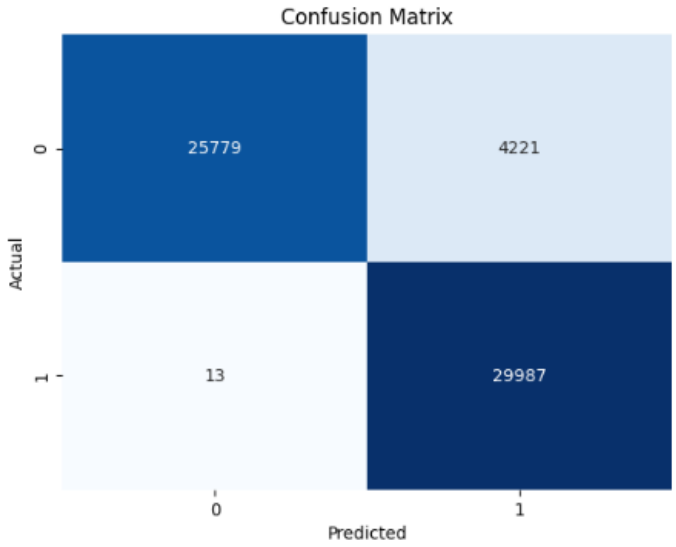
RANDOM FOREST

```
rfmodel = RandomForestClassifier( max_depth=3,n_estimators=3)
rfmodel.fit(X_train,y_train)
y_pred = rfmodel.predict(X_test)
```

```
matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(matrix, annot=True, cbar=False, cmap='Blues', fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Save figure locally
local_graph_path = '/home/ec2-user/environment/graphs/random_forest.png'
fig.write_image(local_graph_path)
s3_key = 'graphs/random_forest.png'
s3_client.upload_file(local_graph_path, bucket_name, s3_key)

print(f" Uploaded graph to s3://{bucket_name}/{s3_key}")
```




```
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.86      0.92     30000
           1       0.88      1.00      0.93     30000

    accuracy                           0.93     60000
   macro avg       0.94      0.93      0.93     60000
weighted avg       0.94      0.93      0.93     60000
```

Gradient Boosting

GRADIENT BOOSTING

```
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
gb = GradientBoostingClassifier(max_depth=2,n_estimators=3)
gb.fit(X_train, y_train)
y_pred = gb.predict(X_test)
```

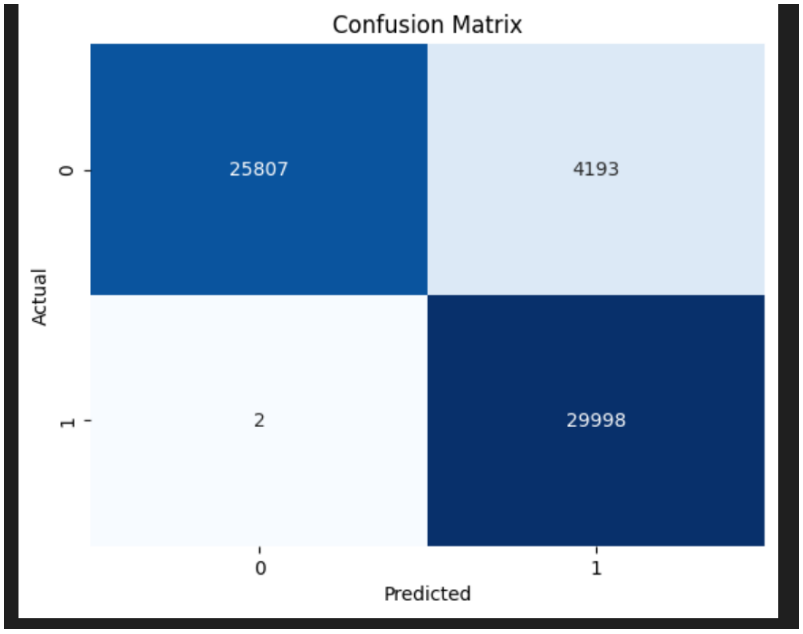
[ ] Python

```
matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(matrix, annot=True, cbar=False, cmap='Blues', fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Save figure locally
local_graph_path = '/home/ec2-user/environment/graphs/gradient_boosting.png'
fig.write_image(local_graph_path)
s3_key = 'graphs/gradient_boosting.png'
s3_client.upload_file(local_graph_path, bucket_name, s3_key)

print(f" Uploaded graph to s3://{bucket_name}/{s3_key}")
```

[ ] Python



```python
#Classification Report
print("Classification Report : ")
print(classification_report(y_test, y_pred))
```

```
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.86      0.92     30000
           1       0.88      1.00      0.93     30000

    accuracy                           0.93     60000
   macro avg       0.94      0.93      0.93     60000
weighted avg       0.94      0.93      0.93     60000
```

Naïve Bayes

NAIVE BAYES

```python
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb.fit(X_train,y_train)
y_pred = nb.predict(X_test)
```

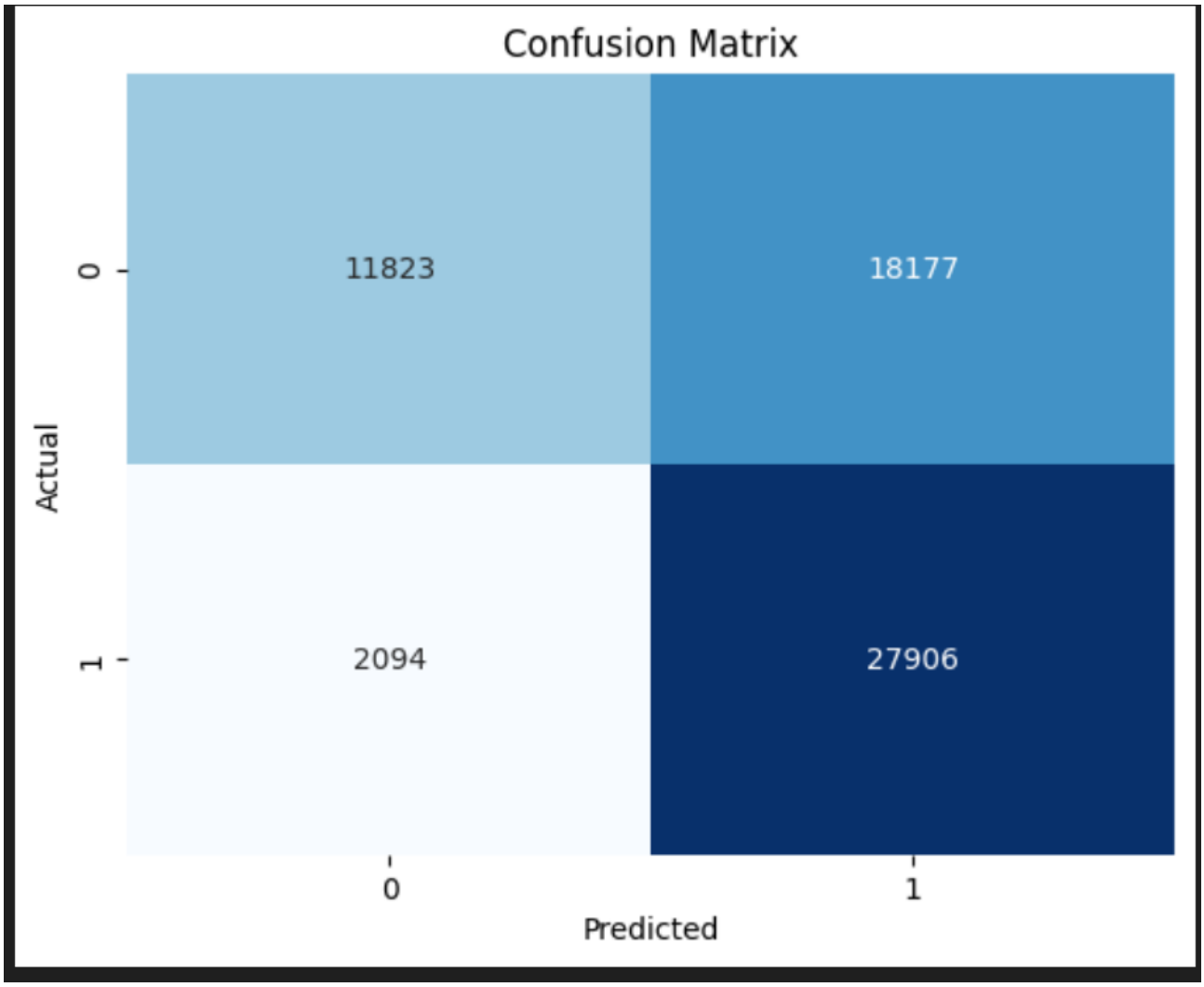
Python

```python
matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(matrix, annot=True, cbar=False, cmap='Blues', fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Save figure locally
local_graph_path = '/home/ec2-user/environment/graphs/naive_bayes.png'
fig.write_image(local_graph_path)
s3_key = 'graphs/naive_bayes.png'
s3_client.upload_file(local_graph_path, bucket_name, s3_key)

print(f" Uploaded graph to s3://{bucket_name}/{s3_key}")
```

Python




```
Classification Report :
              precision    recall  f1-score   support

           0       0.85      0.39      0.54     30000
           1       0.61      0.93      0.73     30000

    accuracy                           0.66     60000
   macro avg       0.73      0.66      0.64     60000
weighted avg       0.73      0.66      0.64     60000
```

Logistic Regression:

LOGISTIC REGRESSION

```
from sklearn.linear_model import Lasso
lasso_clf = LogisticRegression(penalty='l1', solver='liblinear', C=1.0)
lasso_clf.fit(X_train, y_train)
y_pred = lasso_clf.predict(X_test)
```

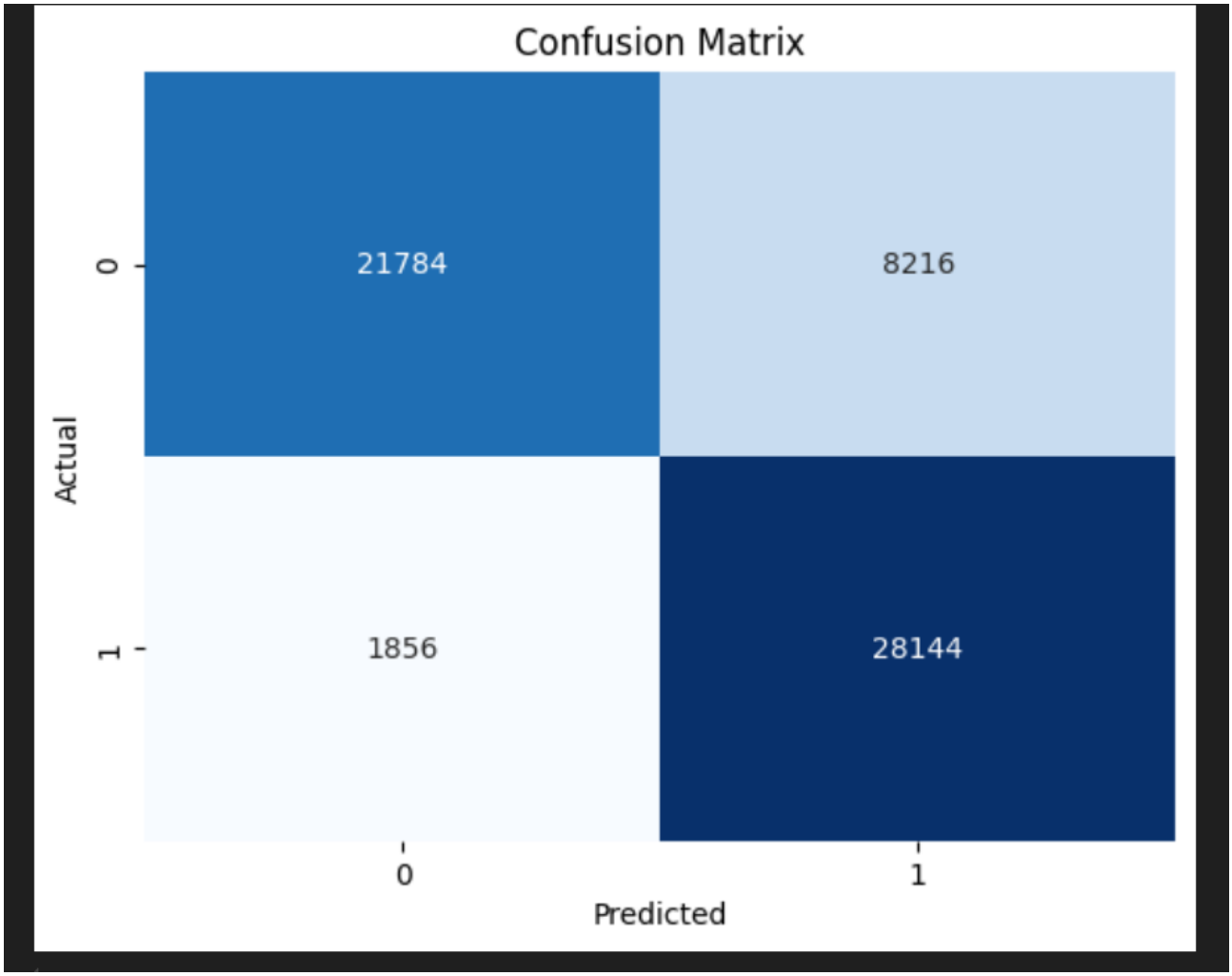
[ ] Python

```
matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(matrix, annot=True, cbar=False, cmap='Blues', fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()


# Save figure locally
local_graph_path = '/home/ec2-user/environment/graphs/logistic_regression.png'
fig.write_image(local_graph_path)
s3_key = 'graphs/logistic_regression.png'
s3_client.upload_file(local_graph_path, bucket_name, s3_key)

print(f" Uploaded graph to s3://{bucket_name}/{s3_key}")
```

[ ] Python



```
...   Classification Report :
                  precision    recall  f1-score   support

               0       0.92      0.73      0.81     30000
               1       0.77      0.94      0.85     30000

        accuracy                           0.83     60000
       macro avg       0.85      0.83      0.83     60000
    weighted avg       0.85      0.83      0.83     60000
```

Superleaner:

```python
def get_models():
    models = list()
    models.append(RandomForestClassifier( max_depth=3))
    models.append(LogisticRegression(penalty='l1', solver='liblinear', C=1.0))
    models.append(GaussianNB())
    return models
```

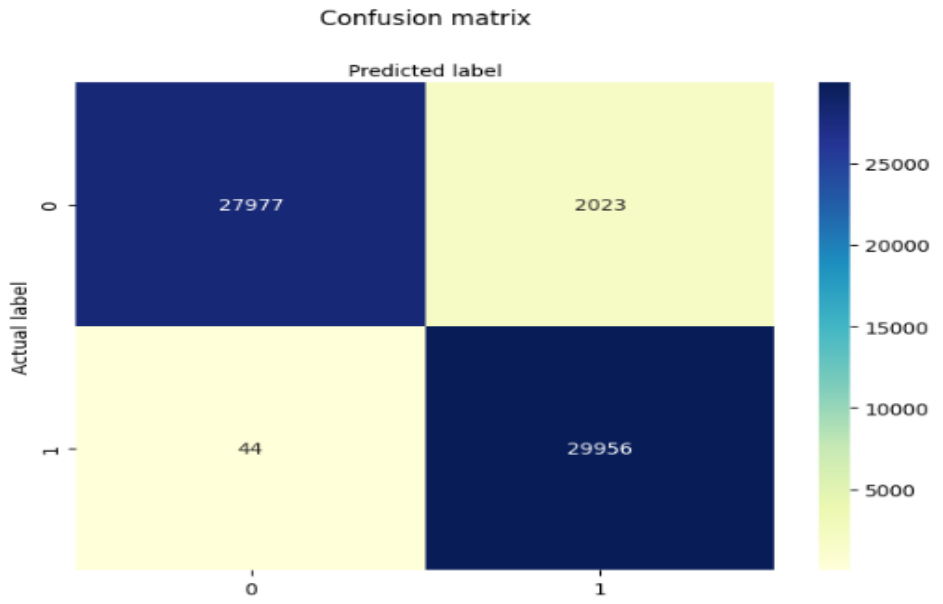
```python
# collect out of fold predictions form k-fold cross validation
def get_out_of_fold_predictions(X, y, models):
    meta_X, meta_y = list(), list()
    # define split of data
    kfold = KFold(n_splits=5, shuffle=True)
    # enumerate splits
    for train_ix, test_ix in kfold.split(X):
        fold_yhats = list()
        # get data
        # Use .iloc to index by integer position
        train_X, test_X = X.iloc[train_ix], X.iloc[test_ix]
        # Assuming y is a numpy array or can be indexed directly by integer position
        train_y, test_y = y[train_ix], y[test_ix]
        meta_y.extend(test_y)
        # fit and make predictions with each sub-model
        for model in models:
            model.fit(train_X, train_y)
            yhat = model.predict(test_X)
            # store columns
            fold_yhats.append(yhat.reshape(len(yhat),1))
        # store fold yhats as columns
        meta_X.append(hstack(fold_yhats))
    return vstack(meta_X), asarray(meta_y)
```

```python
# fit all base models on the training dataset
def fit_base_models(X, y, models):
    for model in models:
        model.fit(X, y)

# fit a meta model
def fit_meta_model(X, y):
    model = GradientBoostingClassifier( max_depth=2,n_estimators=3)
    model.fit(X, y)
    return model

# evaluate a list of models on a dataset
def evaluate_models(X, y, models):
    for model in models:
        yhat = model.predict(X)
        acc = accuracy_score(y, yhat)
        print('%s: %.3f' % (model.__class__.__name__, acc*100))

# make predictions with stacked model
def super_learner_predictions(X, models, meta_model):
    meta_X = list()
    for model in models:
        yhat = model.predict(X)
        meta_X.append(yhat.reshape(len(yhat),1))
    meta_X = hstack(meta_X)
    # predict
    return meta_model.predict(meta_X)
```



Uploaded graph to s3://x23228946-thesis-bucket/graphs/super_learner.png

	precision	recall	f1-score	support
0	1.00	0.93	0.96	30000
1	0.94	1.00	0.97	30000
accuracy			0.97	60000
macro avg	0.97	0.97	0.97	60000
weighted avg	0.97	0.97	0.97	60000